(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 807 891 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
19.11.1997 Bulletin 1997/47

(51) Int. Cl.[6]: **G06F 17/30**

(21) Application number: **97201881.6**

(22) Date of filing: **24.12.1996**

(84) Designated Contracting States:
DE FR GB NL SE

(30) Priority: **11.01.1996 US 583877**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
96203701.6 / 0 784 279

(71) Applicant: **SUN MICROSYSTEMS INC.**
Mountain View, California 94043-1100 (US)

(72) Inventors:
- **Levine, Fredrick E.**
  Boulder, CO 80302 (US)
- **Carter, Bruce A.**
  Los Angeles, CA 90045 (US)

(74) Representative:
Hanna, Peter William Derek et al
Tomkins & Co.,
5 Dartmouth Road
Dublin 6 (IE)

Remarks:
This application was filed on 19 - 06 - 1997 as a divisional application to the application mentioned under INID code 62.

(54) Stateless shopping cart for the web

(57) A shopping cart metaphor is emulated on a network (46) of server (20) and client (35) computing systems. A browser at the client station has a request module to send a shopping page request to the server. A shopping page module in the server sends a shopping page file (40) to the browser in response to the shopping page request. The shopping page file contains items selectable by a user using the browser. A shopping module at the browser generates an add request and sends the add request to the server. This add request contains selected items from the items that were selectable in the shopping page file. A receiver at the server receives the add request from the browser, and a cart list module at the server initialises a shopping cart list. An add module at the server adds the selected items to the shopping cart list. A shopping page module at the server converts the cart list to a cart field, generates a new shopping page file, embeds the cart field in the new shopping page file and sends the new shopping page file to the browser. In this way, the shopping cart field is in a shopping page file that may be managed by the browser at the client station (35).

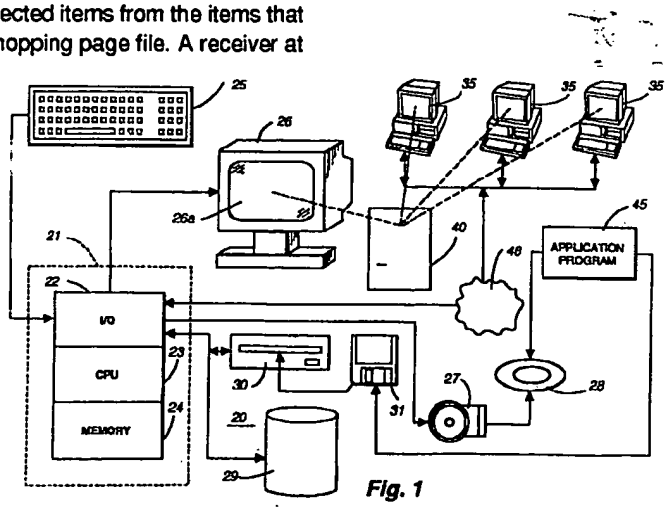

Fig. 1

EP 0 807 891 A1

Description

BACKGROUND OF THE INVENTION

Field of the Invention.

The present invention relates to a client computing station shopping across a network, such as the Internet, for items to be downloaded from a server computing system on the network. More particularly, the invention relates to the server handing off maintenance of a shopping cart metaphor to the client whereby the shopping process at the server is stateless, i.e. no maintenance of the shopping requests by the client.

Description of Prior Art.

Typical database or file-based shopping cart systems require that the user be uniquely identified in order to associate particular data stored on the server with a particular user. This requires the user to log-in or create an account, which is then stored in the server. Each subsequent request from the user must reference the unique identifier, either in the uniform resource locator (URL) or as hidden data passed back through a form submission. Either of these approaches require that the account or ID information of the user be stored on the remote server in the network for some definite period of time. Usually, the user must keep track of the account identifier in order that the prior session information can be retrieved.

There is currently no reliable means to deduce the user's account information from the information accompanying a random request for a page. This is because many web servers, especially fire-walled servers, intentionally do not identify the sender in their data. When the remote server on the network serves the request from the user and stores the user's account information, this serving system must decide how long the data for an individual user will be stored.

In a resource constrained serving system, it is possible that a particular user's account information will be deleted before the user has completed the transaction. For example, a user might start a transaction in the evening, collect several items from the server and then retire for the night leaving the transaction open. Upon returning to the transaction process in the morning, the user is likely to find that the nightly administrative clean up at the remote server has assumed that all transactions were abandoned or completed and has therefore deleted the previous evening's collection of items.

SUMMARY OF THE INVENTION

In accordance with this invention, the above problems of prior shopping cart operations on networks, such as the Internet, have been solved by providing a stateless shopping cart operation at the server. In the stateless shopping cart operation, the server does not maintain a list of selected items for the shopping cart. Rather in each client transaction at the server, the server updates a data field identifying the contents of the shopping cart and sends the updated version of the shopping cart as a data field back to the client.

The stateless shopping cart process is triggered by command or request for a shopping page transmitted from a browser program at a client station to a computer-server. At the server, this request invokes a market application program at the server. In response to the transmitted command, the market program generates and transmits an shopping page file to the browser. The browser builds a shopping page from said shopping page file and sends to the server data strings corresponding to user selected items from the shopping page received and stored by the browser. The server determines if a list containing identification of items previously selected by said user exists, and if not, creates a list identifying items currently selected. This list is returned as a hidden "cart" field in a shopping page file to the browser. This interaction between browser at the client and market program at the server to transmit, receive, add items to the list and return a cart field to the browser continues until terminated by the browser requesting downloading selected items in the cart to the user client station.

The apparatus for emulating a shopping cart metaphor on a network of server and client computing systems has a request module in the browser to send a shopping page request to the server. A shopping page module in the server sends a shopping page file to the browser in response to the shopping page request. The shopping page file contains items selectable by a user using the browser. A shopping module at the browser generates an add request and sends the add request to the server. This add request contains selected items from the items that were selectable in the shopping page file. A receiver at the server receives the add request from the browser, and a cart list module at the server initializes a shopping cart list. An add module at the server adds the selected items to the shopping cart. A shopping page module at the server converts the cart list to a cart field, generates a new shopping page file, embeds the cart field in the new shopping page file and sends the new shopping page file to the browser. In this way, the shopping cart field is in a shopping page file that may be managed by the browser at the client station.

As a further feature of the invention, the shopping module at the browser generates an add request and sends the add request to the server. This add request contains current selected items from the items selectable in the new shopping page file and previously selected items in the cart field. The cart list module at the server converting the cart field of previously selected items to a cart list of previously selected items, and the add module adds the currently selected items from the add request to the cart list. Therefore, the cart list contains previously selected items and the current

and of the market program file at the server. In Internet protocol, this is referred to as the uniform resource locator (URL). The user at the client station using the web browser program will enter in the URL space the desired locator string.

For example, the user might enter the following the string "http:// gemini.west//dx/". In this string, "http:" indicates the string uses the hyper text transfer protocol. The first slash is a separator, and "/gemini.west/" is the location or node address of the server. The "dx" in the string is a directory which may be used along with other directories in the string to identify where a program file resides in the server. To request the market program, the user would type in the string after dx "/cgi-bin/ds.cgi". In this additional string "ds.cgi" is the market program file, and it is located in "cgi-bin". Operation 150 at the browser sends this shopping page request to invoke the market program at the server to send back a shopping page file.

In the server, the shopping page request is received by receiver operation 152. Operation 154 in the server invokes the market program identified by the filename in the request and generates the HTML shopping page file. Once the HTML shopping page file is built by the market program, operation 156 sends this HTML shopping page file to the browser that requested the shopping page.

Back at the browser, operation 158 receives and stores the HTML shopping page file. The shopping page files are stored at the browser to provide a history of pages through which the user may search if desired. In operation 160 this file is built and displayed on the monitor of the browser as a page.

FIG. 3 is illustrative of the HTML shopping page produced from the shopping page file sent by the market program addressed in the shopping page request. Above line 41 in FIG. 3 are selectable icons associated with the browser program. The buttons illustrated in FIG. 3 are examples of display selectable icons provided by the Netscape TM web browser program. Other browser programs might be utilized at the remote client station. Note that the location field 44 indicates the uniform resource locator (URL) previously described in the example of a shopping page request sent by browser 150.

Below line 41 in FIG. 3 is the HTML page produced from the HTML shopping page file returned in response to the shopping page request. The icon "Sun Microsystems," 47A, if selected by the user will bring up the Sun Microsystems, Incorporated home page. Selecting icon SunSoft TM, 47B, will bring up the home page for this subsidiary of Sun Microsystems. Similarly, selecting Products and Solutions, 47C, or Solaris TM Driv r Express, 47D, will bring up the initial screens of those programs.

The shopping page illustrated in FIG. 3 allows the user customer to select one or more of the icons in row 56, labeled 56A through 56F. Each of the icons represents a category of device drivers, selected ones of which the user customer may wish to receive to update a device driver program or to load a device driver program to run a piece of hardware on the user's computer system. The user may s lect software drivers for disk devices, network devices, tape devices, audio devices, video devices or multiple processor modules. Alternatively, the user may enter a device name or number in the blank field 57 and conduct a search for software drivers in this manner. In either event, once the user has requested a list of drivers, another shopping page request will be sent to the server and the server will reply with the requested HTML shopping page file. Decision operation 161 detects a request other than an "add" request, i.e. add items to shopping cart. Since this is not an add request, and operation 161 returns the browser to operation 150 to send the request for the page with a list of drivers. Operations 152, 154 and 156 at the server respond with the requested shopping page file in the same manner as described above.

As shown in FIG. 4, the newly requested shopping page built from the current shopping page file includes a scrollable alphabetical listing 62 of disk device drivers. From this scrollable list, the user may select a disk device driver for his shopping cart. The user may select one or more items from the scrollable window 62. After making a selection the user adds the selected items to his shopping cart by selecting add icon 64. Alternatively, the user may view another page with additional information on the selected drivers by selecting view information icon 65. If the user's request is not an add request, decision operation 161 returns the process to the browser program to detect other user commands. If the user selects the add icon, decision operation 161 branches "Yes" to operation 162. Operation 162 then reads the user selected items on the shopping page, and operation 164 in the browser generates an add request. The add request is a data string including (1) an identification of the market program from which the user is selecting items, (2) a cart field in the shopping page file, if any, and (3) values identifying items just selected by the user which the user wishes to add to its HTML shopping cart, i.e. the cart field.

In FIG. 2B, the browser sends this add request data string to the server in operation 166. At the server, operation 168 receives the add request and invokes the market program in the add request. Operation 170 then separates the field contents in the add request into name value pairs. If there is a cart field in the add request containing previously selected items by the user at the browser, this name i.e. "cart", and value i.e. previous selected items, will be detected at decision operation 172.

Decision operation 172 if it does not find a cart field or does not find a value in the cart field will branch "No" to operation 174. Operation 174 will initialize an empty cart list at the server. In effect, the add request currently being processed in such a situation is a request containing the first choices made by the user at the browser, and there is no prior cart field in the add request.

Accordingly, a cart list must be generated at the server and this operation is performed at step 174.

If there is a value in the cart field received in the add request, then the logical operations in the server branch to convert-to-list module 176. Module 176 converts the cart field string containing previous selected items into a cart list of previous selected items.

In FIG. 2C at operation 178, the server adds selected items just received in the current add request to the cart list produced either at operation 174 or operation 176. Since in our example this is the first collection of selected items in an add request, the cart list will come from operation 174. With the cart list updated to contain the currently selected items in the add request, convert-to-field module 180 converts the cart list at the server to a cart field data string. Operation 182 then generates the HTML shopping page file and inserts the cart field string into that file as a hidden field. Operation 184 sends this HTML shopping page file containing the hidden cart field to the browser. After the HTML shopping page file is sent, there is no need to maintain the cart list at the server. Therefore operation 184 releases the "cart" list temporary storage, and the market program exits.

At the browser, operation 186 receives and stores the HTML shopping page file containing the hidden field with the cart string of previously selected items. From this page file, operation 188 builds and displays the shopping page.

The cart list is not retained by the server. Previously selected items by the user are in the cart field string in the HTML shopping page file. So long as the user maintains the current shopping transaction, or stores the HTML shopping page file, the cart field with selected items is maintained at the browser or in communications between the browser and server.

In our example, it is assumed that the user in the first add request selected one item, a device driver for a disk device where the device driver is denoted by the identifier "AT&T GIS NCR 53C815." Accordingly, the HTML shopping page file received by the browser at operation 186 is in HTML code and looks like the file listed in FIG. 7. Of course, any code consistent with the operating programs could be used. The listing in FIG. 7 is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the European Patent Office patent file or records, but otherwise reserves all copyright rights whatsoever. Section 250 of the listing is an example of a list of selectable items that would appear in window 61 (FIG. 4 and FIG. 5).

The hidden cart field in FIG. 7 is at entry 252 in the file. At entry 252, the hidden field indicates the name of the field is "cart," and the value in the cart field is "AT&T GIS NCR 53C815." In other words, the value is the device driver previously selected by the user.

Operation 188 in the browser when it processes the shopping page file shown in FIG. 7 generates the page shown as FIG. 5 on the user's monitor at the client station. The user may now select additional items, may select another page with other device drivers or may make a review request by selecting icon 67, "Review or download my collection".

Assume that the user selects the network icon 56B. A request for the page containing network device driver selections would go out to the server. An HTML shopping page is returned to the browser from the server. The page is similar to that shown in FIG. 5 except the scrollable items in scrolling window 61 would be network software device drivers. If the user then at operation 200 in FIG. 2C were to select the add icon 64 after selecting a network device driver, decision operation 202 would detect that the request being received was not a review request and would branch the process to operation 164 in FIG. 2A to generate the add request. The browser and server would proceed as described above through operations 166 to 184 to add the selected network device drive to the cart field hidden in the shopping page file sent back to the browser in response to the add request.

Assuming that the user has shopped enough and wishes to review or download the selected collection of drivers, the user would select the icon 67, "Review or download my collection". In this instance, the process would branch yes at decision operation 202 to send review request operation 203 in FIG. 2D. This review request includes the "cart" field containing the list of selected items by the user and the market program identifier.

In Fig. 2D, the review request is received at operation 205 in the server. Operation 207 detects the market program identifier received in the review request and invokes the market program. Operation 209 converts the "cart" field in the review request to a "cart" list of selected items. At this point the server could add or delete items in the cart list; however in the review request there is no add or delete process so operation 211 in Fig. 2E converts the "cart" list back to the "cart" field.

In Fig. 2E, operation 213 then generates a HTML cart page file and inserts the cart field string into that file. Operation 215 sends this HTML cart page file containing the cart field to the browser. After the HTML cart page file is sent, operation 217 releases the "cart" list temporary storage, and the market program exits.

At the browser, operation 219 receives and stores the HTML cart page file containing the cart field with the string of collected items selected by the user. From this page file, operation 221 builds and displays the cart page. The cart page includes a window 91 containing the list of collected items selected by the user. The cart page is shown in Fig. 6. If the user selects a view information icon 65 in Fig. 6, then a request will go to the server to produce an HTML shopping page file with information about the selected items in the cart field. If the user wishes to delete one or more items from the collected items in window 91, the user selects the items

to be deleted by highlighting them and then selects button 93.

In Fig. 2E, at operation 223 the user select a delete item in window 91 and then selects delete icon 93 as discussed above. Decision operation 225 tests whether the operation performed by the user was a delete request or a download request. Since the operation was a delete request, the process branches "No" to delete request generation module 227. Module 227 generates a delete request containing (1) the market program identifier, (2) the "cart" field of collected items and (3) the selected delete items to be deleted from the cart field.

The delete request is sent by operation 229 in Fig. 2F from the browser to the server. At the server operation 231 receives the delete request and invokes the market program. Operation 233 then converts the cart field string in the delete request into a cart list of collected items previously selected by the user. Operation 225 reads the selected delete items in the delete request and deletes these selected delete items from the cart list.

The updated cart list is converted back to a cart list by operation 211 in Fig. 2E. The server again builds the cart page file at operation 213 and sends the cart page file with updated cart field at operation 215 back to the browser. Receive and store module 219 receives the cart page and stores it at the browser. Operation 221 again builds the cart page and displays the cart page with the updated listed of collected items, i.e. the previous list of collected items less the selected delete items just deleted. Assuming the user is now ready to download the remaining items in the shopping cart, the user at operation 223 in Fig 2E selects download icon 94 (Fig. 6). Decision operation 225 detects the presence of a download request and the process branches to operation 204 in Fig. 2G.

In FIG. 2G, operation 204 at the browser sends the download request to the server. The download request identifies the market program and thereby the server providing the market program and includes the cart field with the collection of selected items to be downloaded.

At the server, receive operation 206 receives the download request from the browser, and operation 208 invokes the market program identified in the download request. Conversion operation 210 at the server converts the cart field string into a cart list of selected items. Retrieve and send operation 212 then retrieves each of the device drivers identified as a selected item on the cart list and sends each identified device driver to the browser. At the browser, operation 214 receives the selected software device drivers and loads them into a storage device for use by the user.

During this shopping process, when a HTML shopping page file is received at the browser, this file is stored. The browser provides a facility whereby a history of HTML page files may be reviewed. This permits the user to browse to other pages not containing the cart field but at a later time to return to the page containing the cart field and resume shopping. Because the cart data, i.e. the selected items in the cart, are in the HTML shopping page file, the user can store that file for use at a later time. In this case, a later time might be another day when accessing the same server and market program over the Internet.

Thus, the present invention provides a computer-executable process embedded in a computer-useable medium, for example, a storage medium containing a computer program, for shopping on the Internet or any network in which all information necessary for meeting the user's item selections are kept in hidden fields in pages that may be retained by the user at the client station. The server need not retain any information about who the user was or what the user selected.

While the invention has been particularly shown and described with reference to a preferred embodiment thereof, it will be understood by those skilled in the art that various other changes in the form and details may be made therein without departing from the spirit and scope of the invention.

In the claims which follow, we make no claim to any particular source code, object code, or computer program listing as such, within the meaning of Article 52(3) of the European Patent Convention.

Claims

1. A computer-executable process, embedded in a computer-useable medium, for supplying items on a network (46), the network having at least one computer-server (20) for communicating with users employing a browser program on a terminal/computer (35) at a location remote from said computer-server, said embedded process comprising the steps of:

   receiving (152), at the computer-server, a transmitted command from said browser program for a shopping page (40);

   in response to said transmitted command, generating (154) a shopping page file and transmitting (156) the shopping page file to said browser program;

   receiving (168), at the computer-server, at least one user selected item from the shopping page received (158) by the browser program;

   creating (174) a list at the computer server;

   at the computer server, adding (178) to the list each user selected item received by said receiving step;

   returning (184) from the computer server the list of items in an entry of a shopping page file to said browser program; and

continuing user selection (200), receiving (168) data strings, adding (178) items to said list and returning (184) said list until termination by the user.

2. An embedded process in accordance with claim 1, further comprising the step of:

sending (215) from the computer-server selected items to the user in response to a command initiated (203) by the browser program.

3. An embedded process in accordance with claim 2, further comprising the step of:

at the computer server, deleting (235) from the list each user selected item received by said receiving step.

4. An embedded process in accordance with claim 1, wherein said user selected items identify computer programs, and including at the computer-server the steps of:

downloading, upon receipt of a command (204) from the browser, computer programs identified by the user selected programs.

5. An embedded process in accordance with claim 1, wherein said creating (174) step comprises the steps of:

separating (170) fields in the data strings into name/value pairs;

determining (172) if there are any values in a "cart" field by examination of the "cart" field, and if so,

converting (176) said values into a list of previously selected items.

6. An embedded process in accordance with claim 5, wherein said adding step comprises the steps of:

adding new selected items to said list;

converting (180) said list to a field data string;

generating (182) a new shopping page file with said field data string in a hidden field thereof, and

transmitting (184) said new shopping page to said browser program.

7. A computer-executable process according to any of claims 1 to 6, characterized in that the computer-useable medium is a storage medium.

8. A computer-executable process according to claim 7, stored in a memory, a hard disk, a floppy disk, or a CD-ROM.

9. A computer-executable process according to any of claims 1 to 6, characterized in that the computer-useable medium is a transmission medium, or a communications network, preferably the Internet.

10. A computer program for executing a computer process in a server computer (20) for responding to shopping requests from a browser system at a client station (35) so that a shopping cart collection of items selected at the browser system is managed at the browser system rather than the server, said program comprising;

a first computer code mechanism (168) for receiving an add request from the browser system, said add request identifying a market system at the server and having a shopping cart field and a selected item to be added by the market system to a shopping cart list;

a second computer code mechanism (176) for converting a shopping cart field in the add request to the shopping cart list;

a third computer code mechanism (178) for adding the selected item to the shopping cart list to provide an updated cart list containing the selected item and any previous selected items from earlier add requests;

a fourth computer code mechanism (180) for converting the updated cart list to an updated version of the shopping cart field;

a fifth computer code mechanism (182) for generating a page file and embedding the updated version of the shopping cart field in the page file; and,

a sixth computer code mechanism (184) for sending the page file to the browser system at the client station whereby the page file, containing one or more selected items in the shopping cart field, is managed by the browser system at the client station.

11. A computer program according to claim 10, further comprising,

a seventh computer code mechanism (185) for releasing storage space used to store the shopping cart list at the server.

12. A computer program according to claim 11, further comprising,

an eighth computer code mechanism (206) for receiving a download request from the browser system, said download request identifying a market system at the server and having a shopping cart field of previously selected items;

a ninth computer code mechanism (210) for converting the shopping cart field of previously selected items from a download request to a shopping cart list of downloadable selected items; and,

a tenth computer code mechanism (212) for retrieving each selected computer file identified from the shopping cart list of downloadable selected items and sending the selected computer file to the browser system at the client.

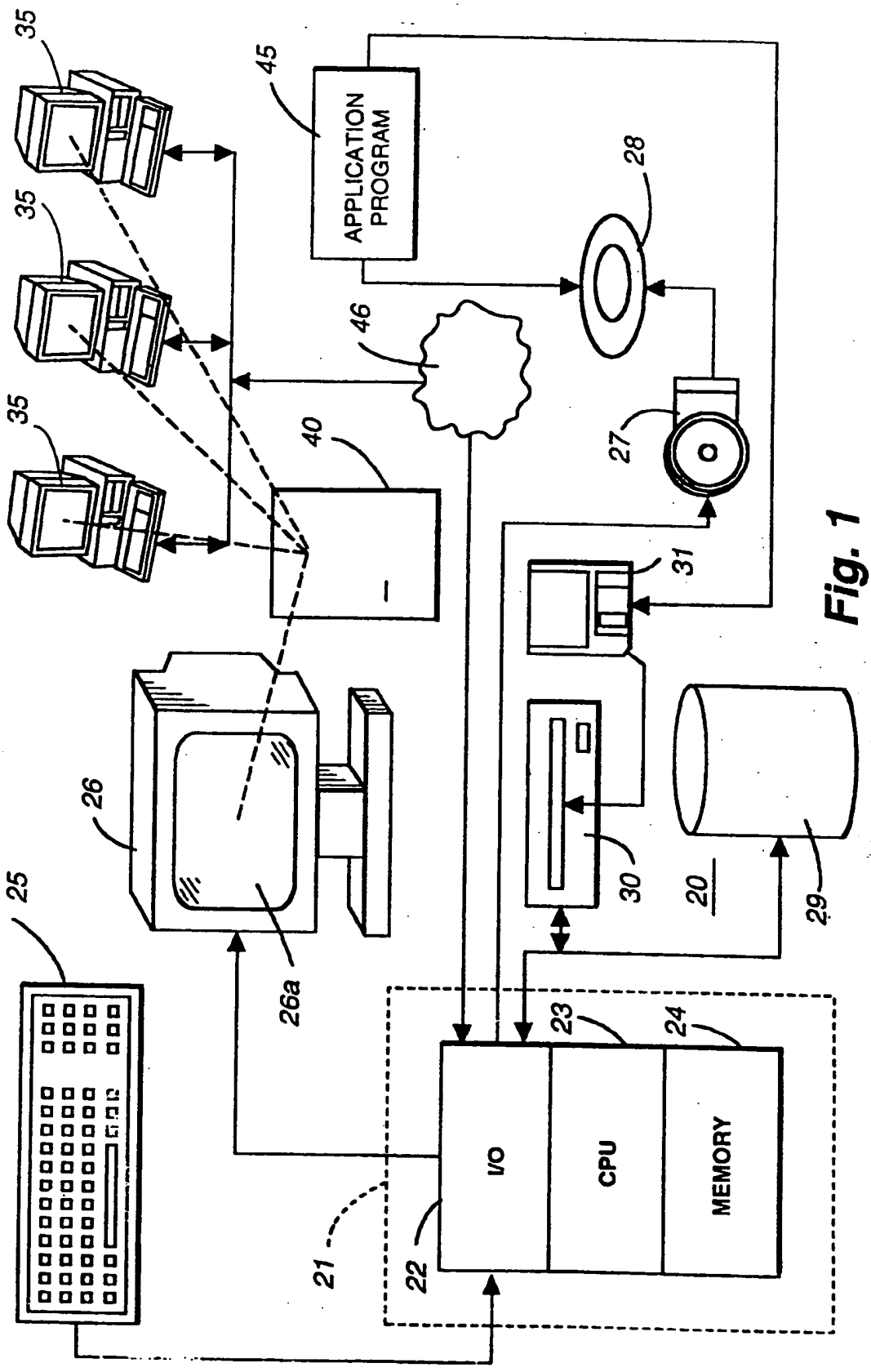
35
35
35
45
APPLICATION PROGRAM
28
46
40
27
31
26
30
20
29
25
26a
23
24
I/O
CPU
MEMORY
21
22

Fig. 1

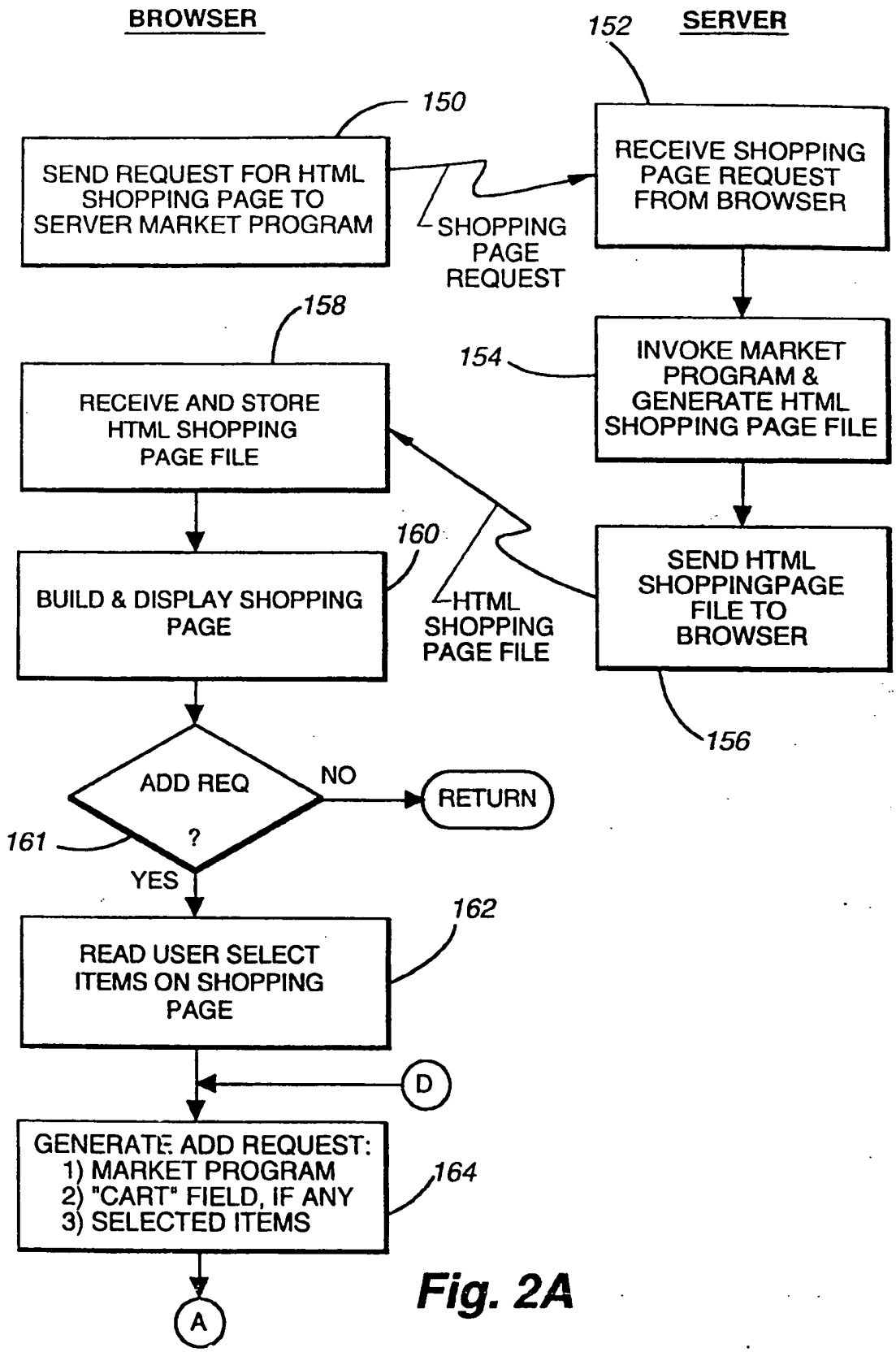
BROWSER
SERVER
152
150
SEND REQUEST FOR HTML SHOPPING PAGE TO SERVER MARKET PROGRAM
SHOPPING PAGE REQUEST
RECEIVE SHOPPING PAGE REQUEST FROM BROWSER
158
154
RECEIVE AND STORE HTML SHOPPING PAGE FILE
INVOKE MARKET PROGRAM & GENERATE HTML SHOPPING PAGE FILE
160
BUILD & DISPLAY SHOPPING PAGE
HTML SHOPPING PAGE FILE
SEND HTML SHOPPINGPAGE FILE TO BROWSER
156
ADD REQ ?
NO
RETURN
161
YES
162
READ USER SELECT ITEMS ON SHOPPING PAGE
D
GENERATE ADD REQUEST:
1) MARKET PROGRAM
2) "CART" FIELD, IF ANY
3) SELECTED ITEMS
164
A

Fig. 2A

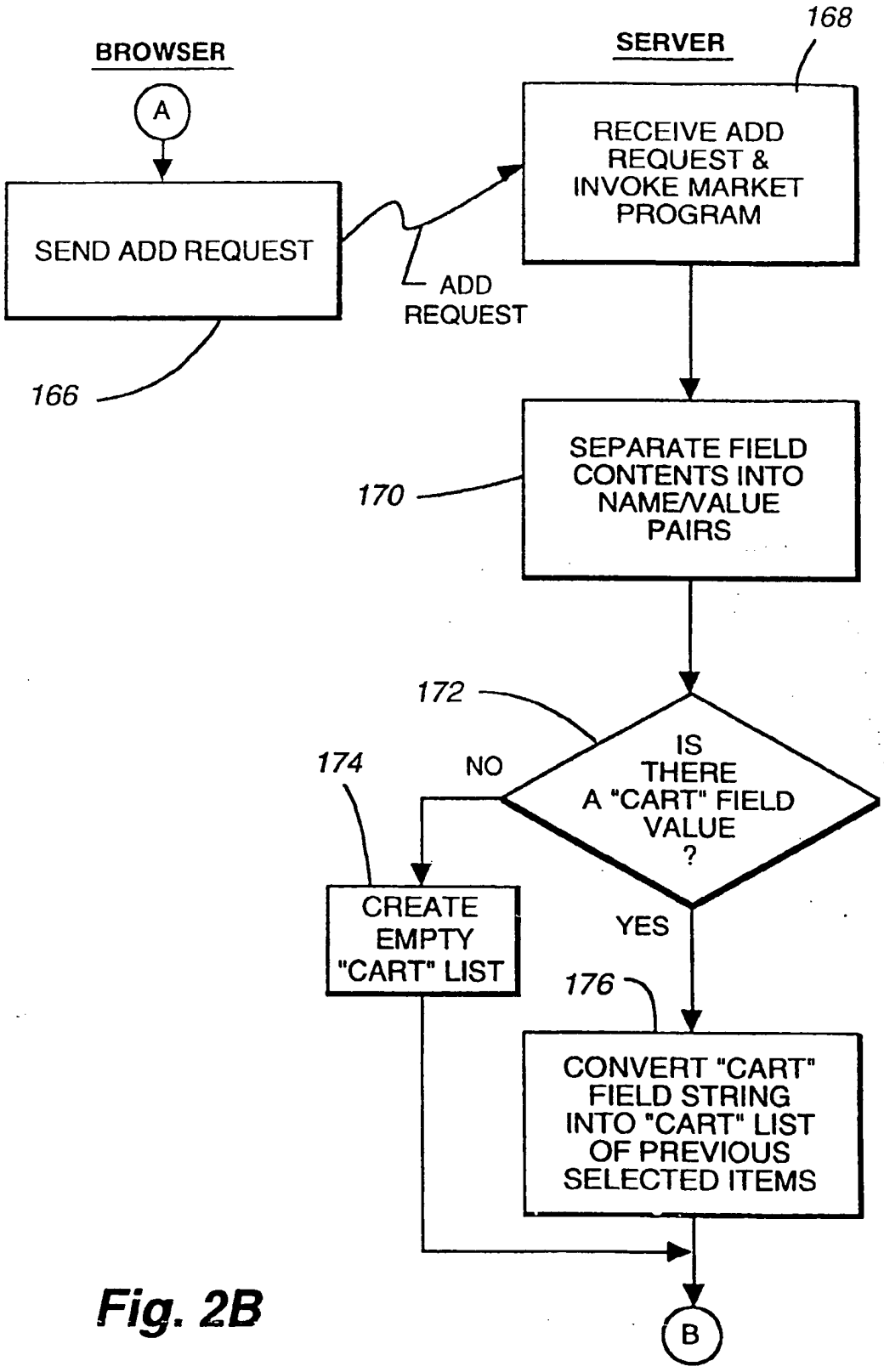
BROWSER
SERVER
168
A
RECEIVE ADD REQUEST & INVOKE MARKET PROGRAM
SEND ADD REQUEST
ADD REQUEST
166
SEPARATE FIELD CONTENTS INTO NAME/VALUE PAIRS
170
172
IS THERE A "CART" FIELD VALUE ?
NO
174
YES
CREATE EMPTY "CART" LIST
176
CONVERT "CART" FIELD STRING INTO "CART" LIST OF PREVIOUS SELECTED ITEMS
B

Fig. 2B

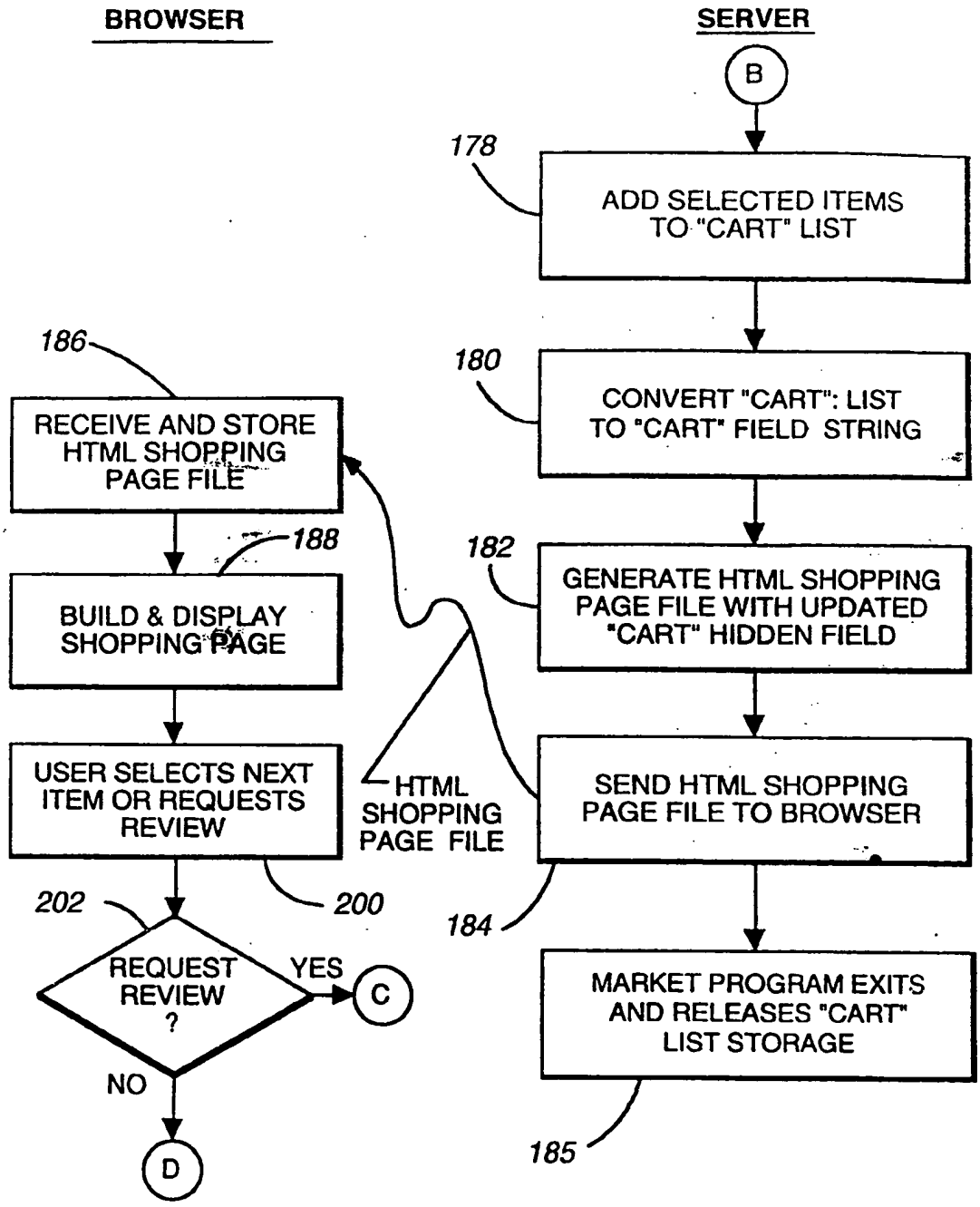
BROWSER
SERVER
B
178
ADD SELECTED ITEMS TO "CART" LIST
180
CONVERT "CART": LIST TO "CART" FIELD STRING
182
GENERATE HTML SHOPPING PAGE FILE WITH UPDATED "CART" HIDDEN FIELD
SEND HTML SHOPPING PAGE FILE TO BROWSER
184
MARKET PROGRAM EXITS AND RELEASES "CART" LIST STORAGE
185
186
RECEIVE AND STORE HTML SHOPPING PAGE FILE
188
BUILD & DISPLAY SHOPPING PAGE
USER SELECTS NEXT ITEM OR REQUESTS REVIEW
200
HTML SHOPPING PAGE FILE
202
REQUEST REVIEW ?
YES
C
NO
D

Fig. 2C

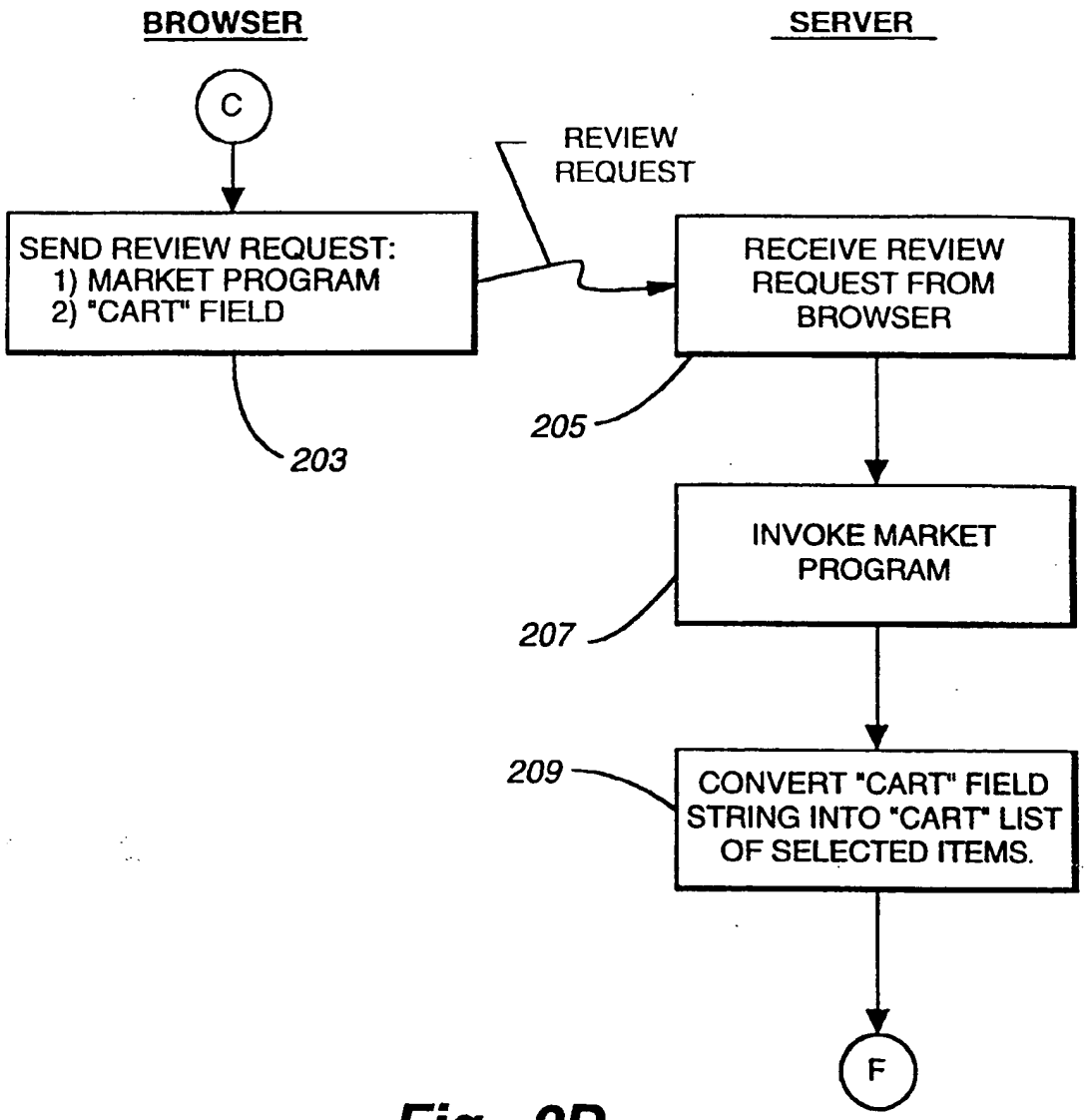
BROWSER
SERVER
C
REVIEW REQUEST
SEND REVIEW REQUEST:
1) MARKET PROGRAM
2) "CART" FIELD
RECEIVE REVIEW REQUEST FROM BROWSER
203
205
INVOKE MARKET PROGRAM
207
209
CONVERT "CART" FIELD STRING INTO "CART" LIST OF SELECTED ITEMS.
F

Fig . 2D

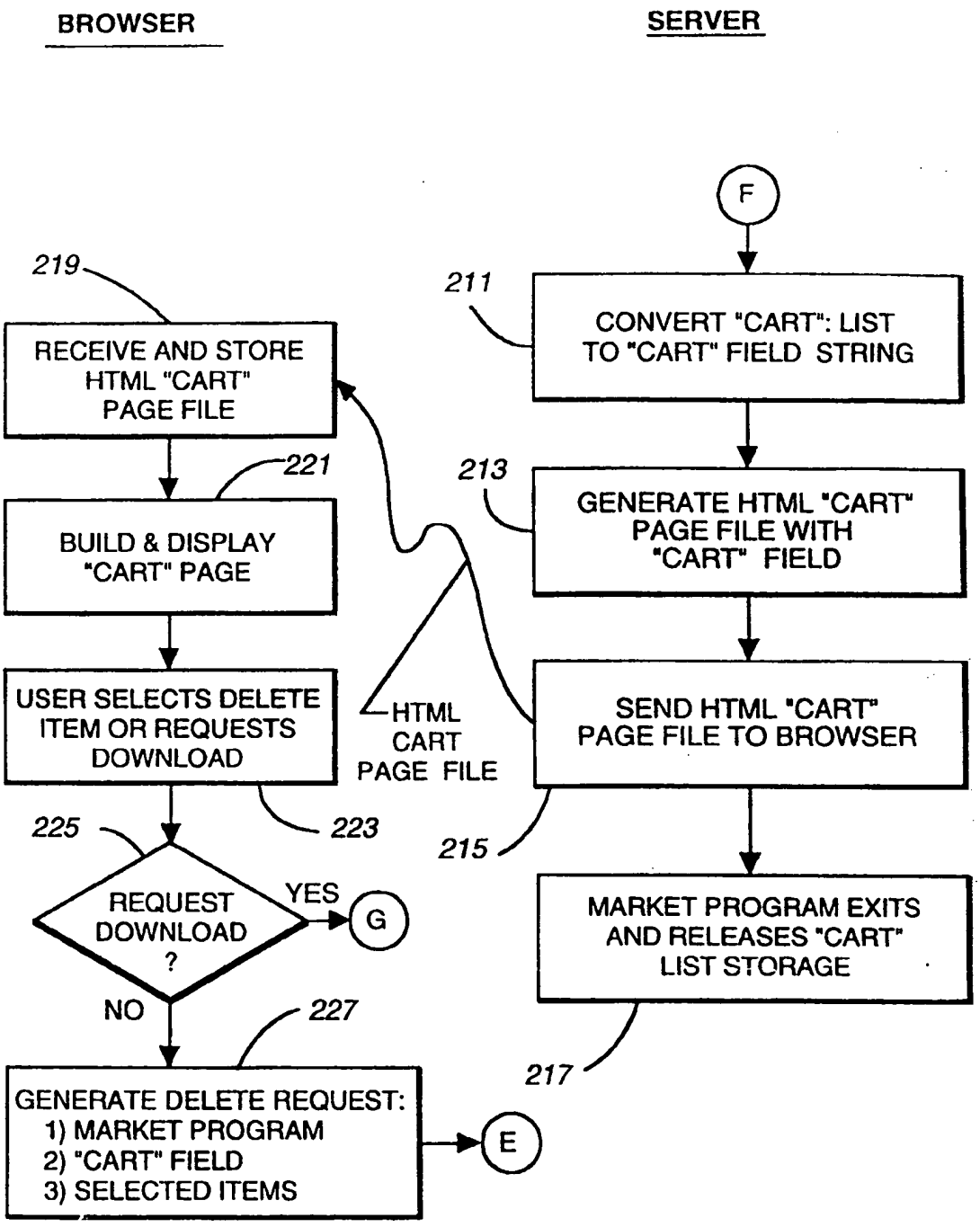
BROWSER
SERVER
F
211
CONVERT "CART": LIST TO "CART" FIELD STRING
213
GENERATE HTML "CART" PAGE FILE WITH "CART" FIELD
SEND HTML "CART" PAGE FILE TO BROWSER
215
MARKET PROGRAM EXITS AND RELEASES "CART" LIST STORAGE
217
HTML CART PAGE FILE
219
RECEIVE AND STORE HTML "CART" PAGE FILE
221
BUILD & DISPLAY "CART" PAGE
USER SELECTS DELETE ITEM OR REQUESTS DOWNLOAD
223
225
REQUEST DOWNLOAD ?
YES
G
NO
227
GENERATE DELETE REQUEST:
1) MARKET PROGRAM
2) "CART" FIELD
3) SELECTED ITEMS
E

Fig. 2E

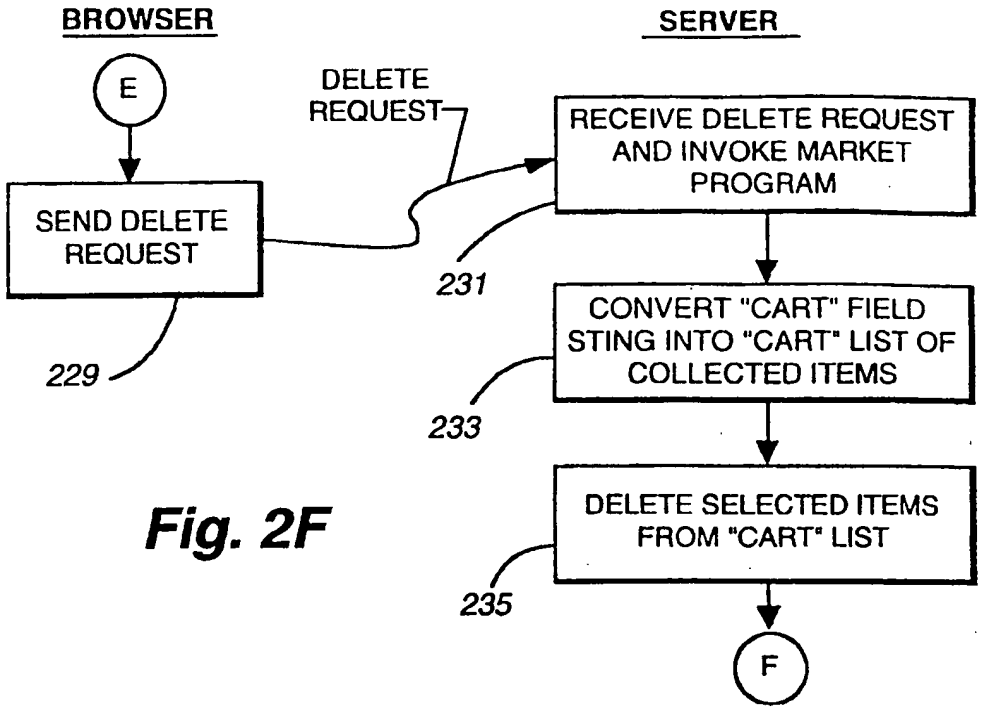
BROWSER
SERVER
E
DELETE REQUEST
SEND DELETE REQUEST
RECEIVE DELETE REQUEST AND INVOKE MARKET PROGRAM
231
229
CONVERT "CART" FIELD STING INTO "CART" LIST OF COLLECTED ITEMS
233
DELETE SELECTED ITEMS FROM "CART" LIST
235
F

Fig. 2F

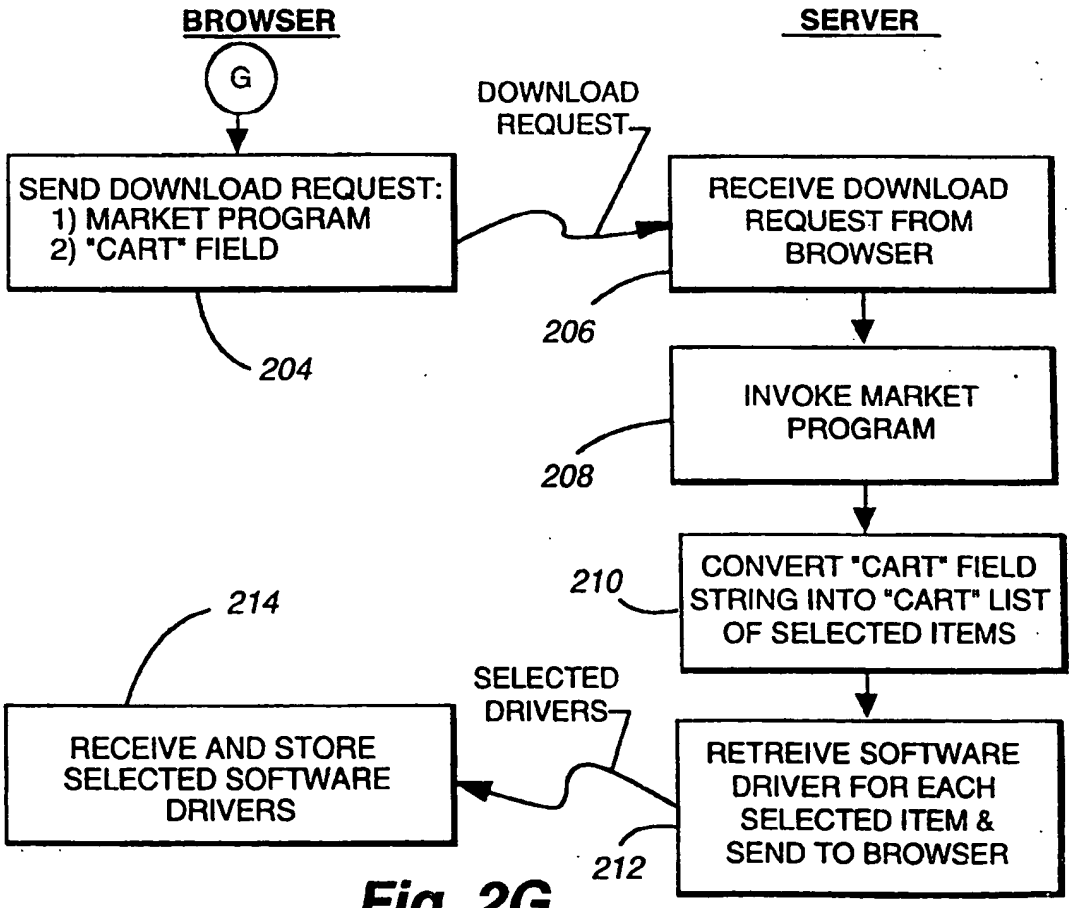
BROWSER
SERVER
G
DOWNLOAD REQUEST
SEND DOWNLOAD REQUEST:
1) MARKET PROGRAM
2) "CART" FIELD
RECEIVE DOWNLOAD REQUEST FROM BROWSER
204
206
INVOKE MARKET PROGRAM
208
CONVERT "CART" FIELD STRING INTO "CART" LIST OF SELECTED ITEMS
210
214
SELECTED DRIVERS
RECEIVE AND STORE SELECTED SOFTWARE DRIVERS
RETREIVE SOFTWARE DRIVER FOR EACH SELECTED ITEM & SEND TO BROWSER
212

Fig. 2G

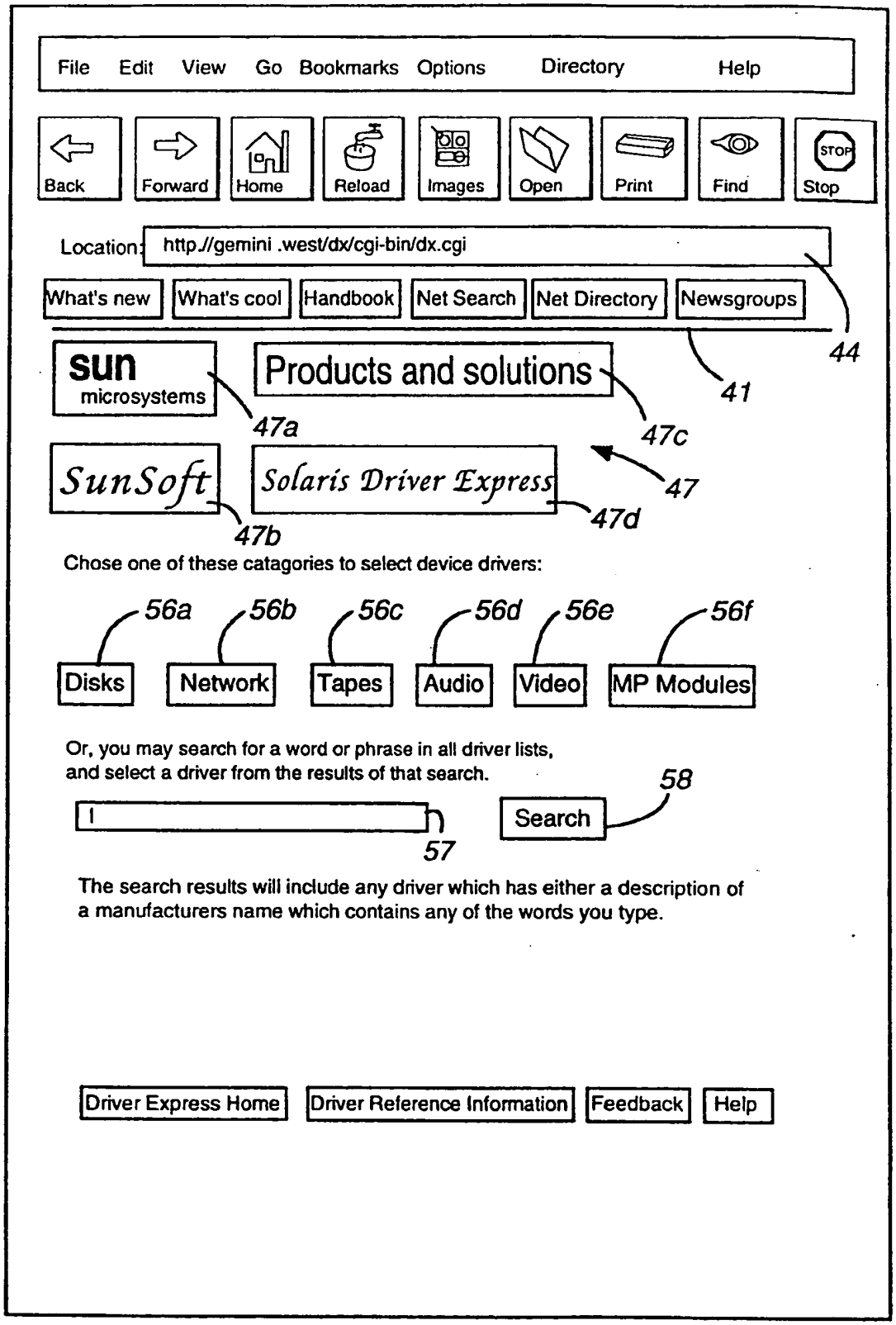
File Edit View Go Bookmarks Options Directory Help
Back
Forward
Home
Reload
Images
Open
Print
Find
Stop
Location: http://gemini .west/dx/cgi-bin/dx.cgi
What's new
What's cool
Handbook
Net Search
Net Directory
Newsgroups
44
41
sun microsystems
47a
Products and solutions
47c
47
SunSoft
47b
Solaris Driver Express
47d
Chose one of these catagories to select device drivers:
56a
56b
56c
56d
56e
56f
Disks
Network
Tapes
Audio
Video
MP Modules
Or, you may search for a word or phrase in all driver lists,
and select a driver from the results of that search.
58
Search
57
The search results will include any driver which has either a description of
a manufacturers name which contains any of the words you type.
Driver Express Home
Driver Reference Information
Feedback
Help

Fig. 3

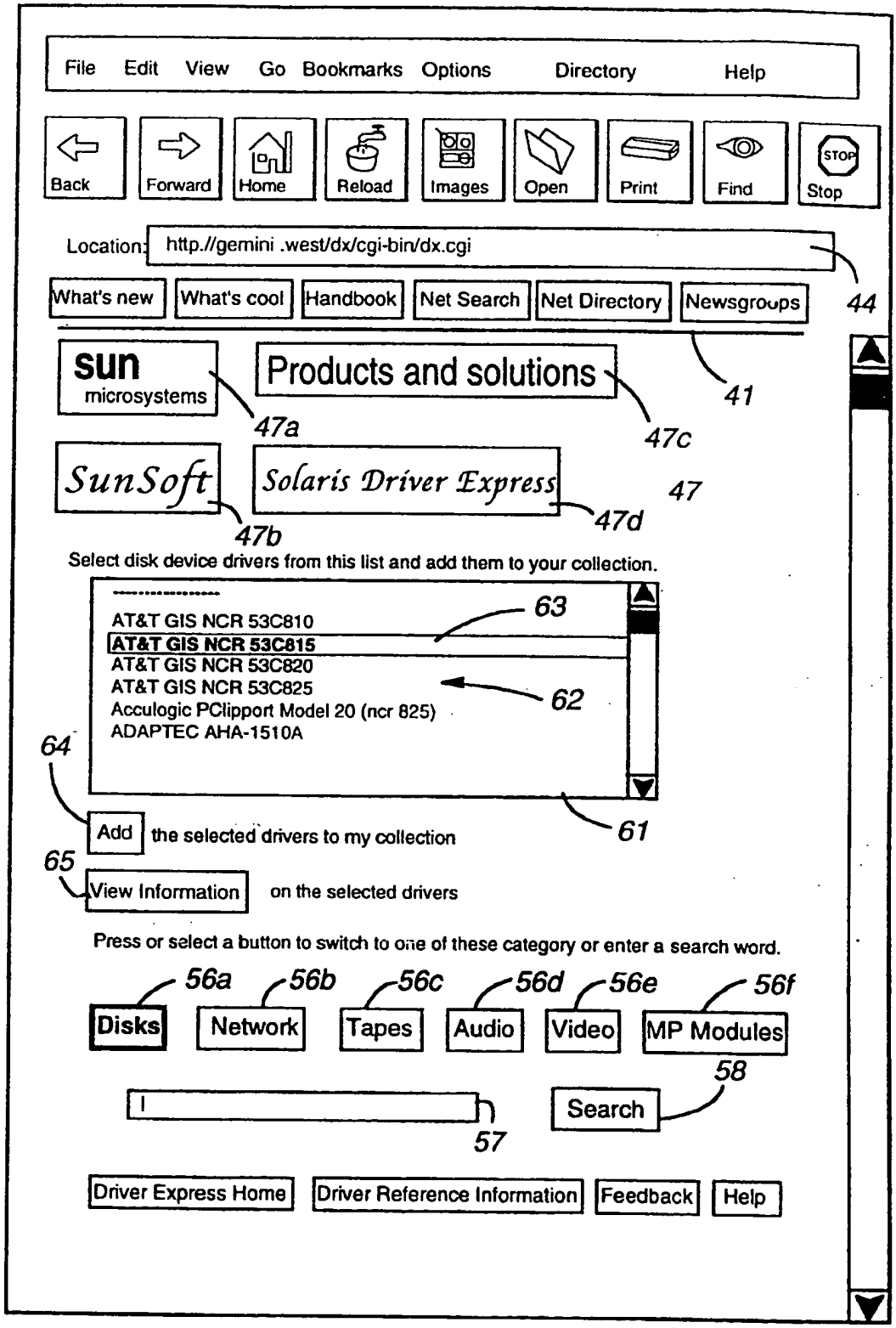
File Edit View Go Bookmarks Options Directory Help
Back
Forward
Home
Reload
Images
Open
Print
Find
Stop
Location: http://gemini .west/dx/cgi-bin/dx.cgi
44
What's new
What's cool
Handbook
Net Search
Net Directory
Newsgroups
41
sun microsystems
47a
Products and solutions
47c
SunSoft
47b
Solaris Driver Express
47d
47
Select disk device drivers from this list and add them to your collection.
AT&T GIS NCR 53C810
AT&T GIS NCR 53C815
AT&T GIS NCR 53C820
AT&T GIS NCR 53C825
Acculogic PClipport Model 20 (ncr 825)
ADAPTEC AHA-1510A
63
62
64
61
Add the selected drivers to my collection
65
View Information on the selected drivers
Press or select a button to switch to one of these category or enter a search word.
56a
56b
56c
56d
56e
56f
Disks
Network
Tapes
Audio
Video
MP Modules
58
Search
57
Driver Express Home
Driver Reference Information
Feedback
Help

Fig. 4

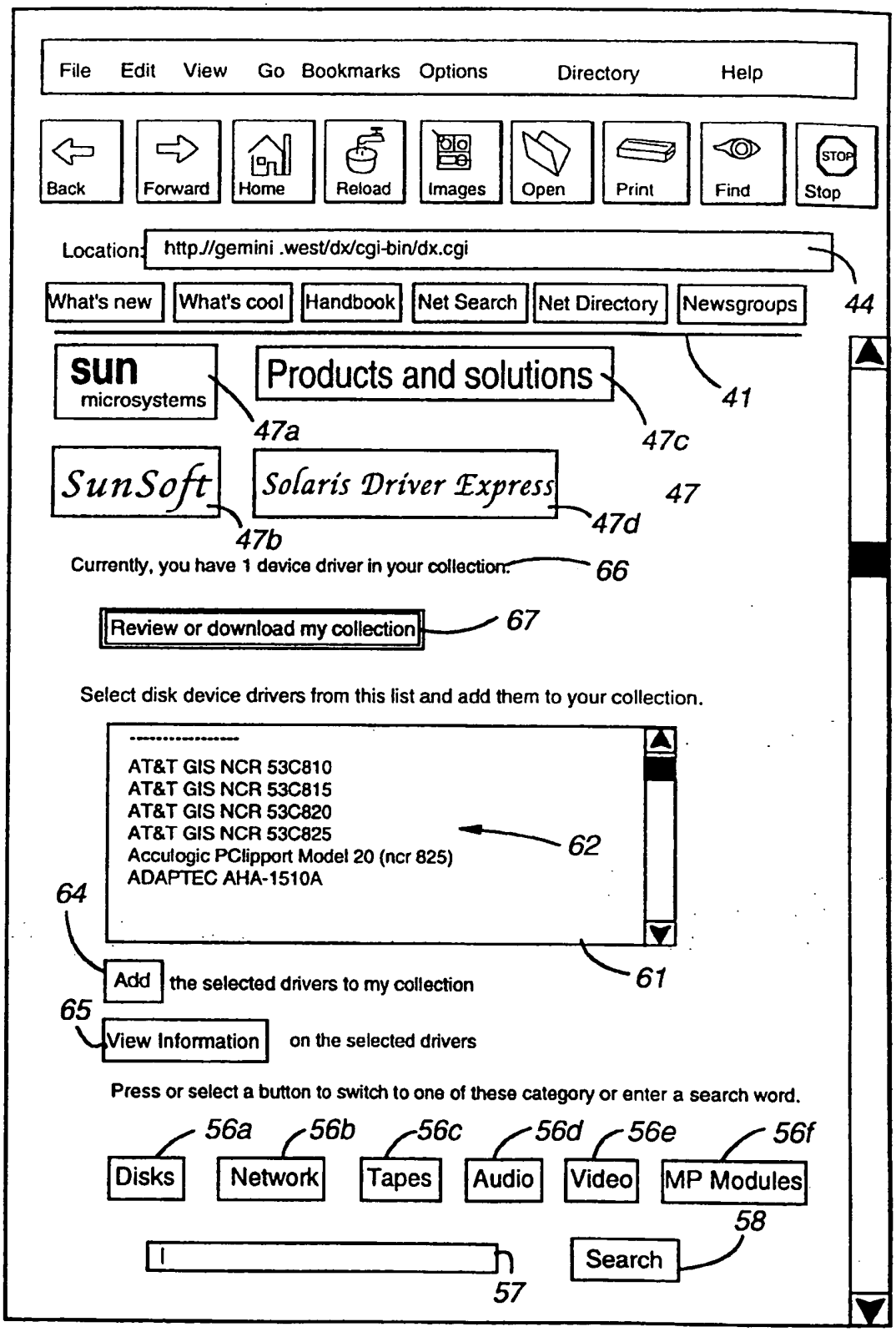
File Edit View Go Bookmarks Options Directory Help
Back
Forward
Home
Reload
Images
Open
Print
Find
Stop
Location: http://gemini .west/dx/cgi-bin/dx.cgi
44
What's new
What's cool
Handbook
Net Search
Net Directory
Newsgroups
41
sun microsystems
47a
Products and solutions
47c
SunSoft
47b
Solaris Driver Express
47d
47
Currently, you have 1 device driver in your collection:
66
Review or download my collection
67
Select disk device drivers from this list and add them to your collection.
AT&T GIS NCR 53C810
AT&T GIS NCR 53C815
AT&T GIS NCR 53C820
AT&T GIS NCR 53C825
Acculogic PClipport Model 20 (ncr 825)
ADAPTEC AHA-1510A
62
64
61
Add the selected drivers to my collection
65
View Information on the selected drivers
Press or select a button to switch to one of these category or enter a search word.
56a
56b
56c
56d
56e
56f
Disks
Network
Tapes
Audio
Video
MP Modules
58
Search
57

Fig. 5

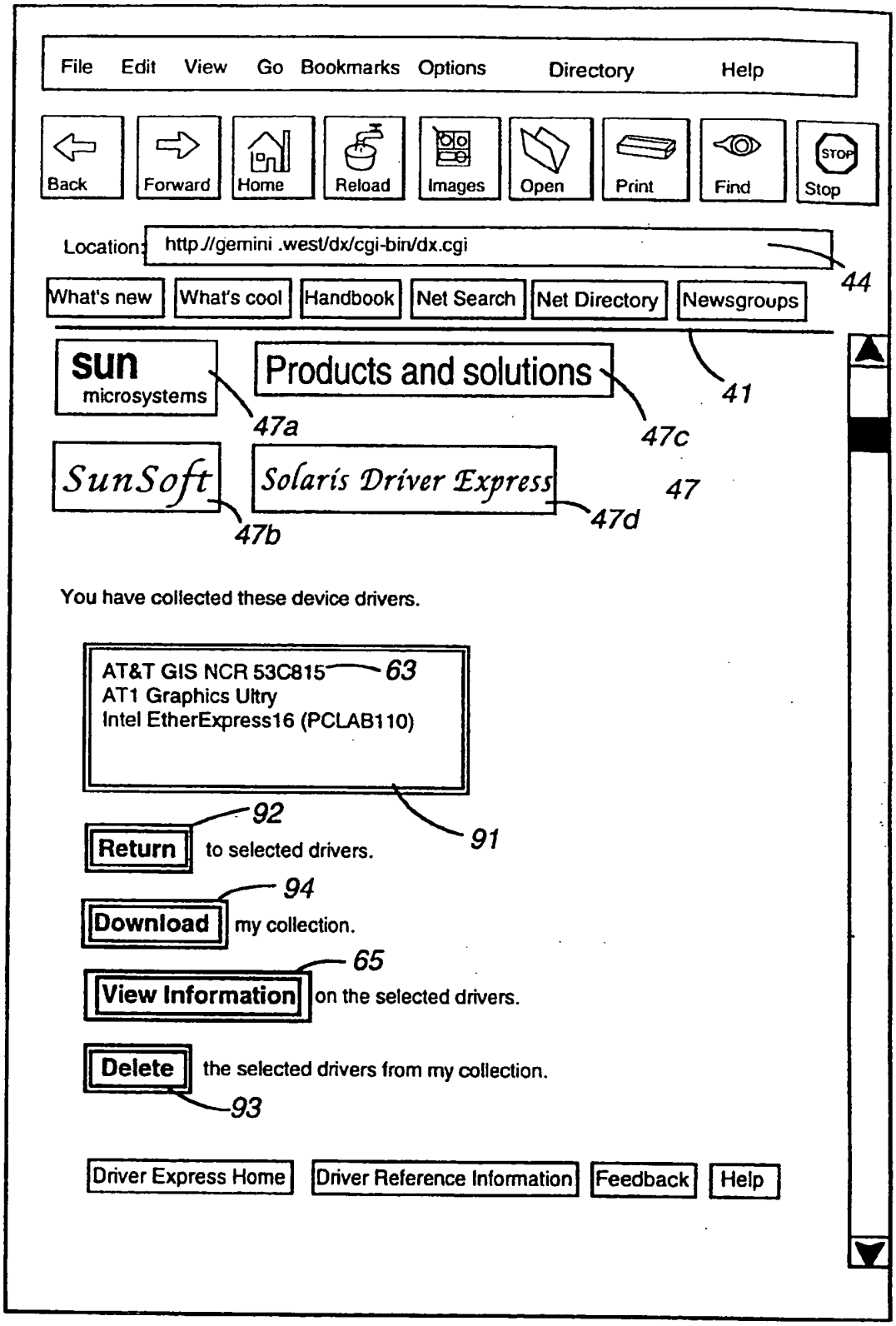
File Edit View Go Bookmarks Options Directory Help
Back
Forward
Home
Reload
Images
Open
Print
Find
Stop
Location: http://gemini.west/dx/cgi-bin/dx.cgi
44
What's new
What's cool
Handbook
Net Search
Net Directory
Newsgroups
41
sun microsystems
47a
Products and solutions
47c
SunSoft
47b
Solaris Driver Express
47d
47
You have collected these device drivers.
AT&T GIS NCR 53C815
63
AT1 Graphics Ultry
Intel EtherExpress16 (PCLAB110)
91
92
Return
to selected drivers.
94
Download
my collection.
65
View Information
on the selected drivers.
Delete
the selected drivers from my collection.
93
Driver Express Home
Driver Reference Information
Feedback
Help

Fig. 6

Fig. 7A

```
<HTML>
<HEAD>
<TITLE>Driver Selection-MP Modules</TITLE>
</HEAD>
<BODY>
<A HREF="http://www.sun.com:80/navbar1.products-n-solutions.lightpurple.580x43">
<IMG SRC="/dx/navbar1.products-n-solutions.lightpurple.580x43.qif"></A>
<BR>
<A HREF="/cgi-bin/imagemap/navbr2.sunsoft-driverexppess.aqua580x35">
<IMG SCR="/dx/images/navbr2.sunsoft-driverexpress.aqua.580x35.gif"></A>

<FORM METHOD-"POST" ACTION="/dx-cig-bin/dx.cgi">

Currently, you have <B>3 device drivers </B> in your collection.

<P>
<OL>
    <INPUT TYPE="submit" NAME= "which_button"
                VALUE="Review or doneload my collection">
</OL>
<P>
<HR>
<0L>
    Select <B>MP module</B> drivers from this list and add
    them to your collection:
```

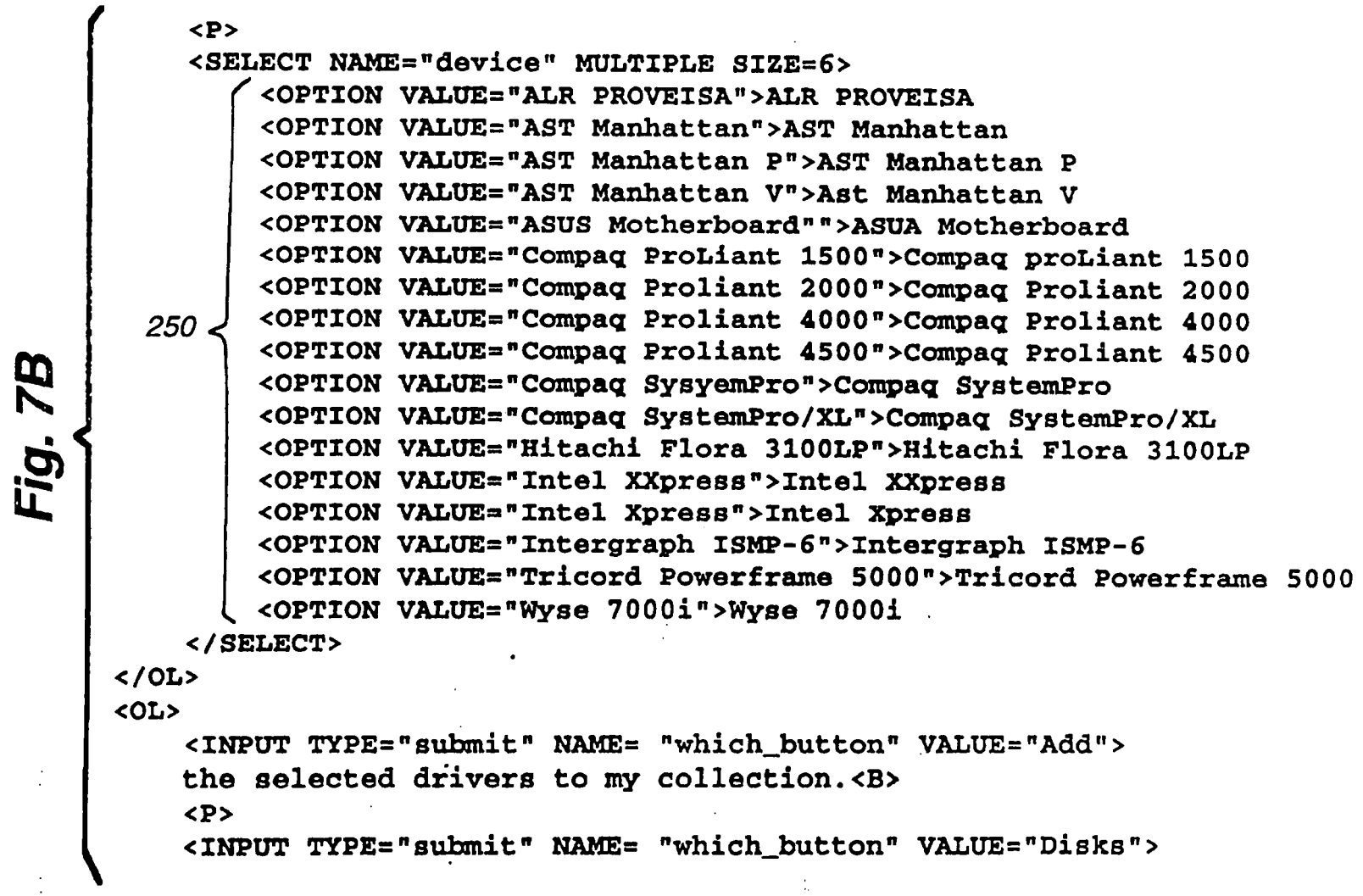

Fig. 7B

EP 0 807 891 A1

Fig. 7C

```
    <INPUT TYPE="submit" NAME= "which_button" VALUE="Networks">
    <INPUT TYPE="submit" NAME= "which_button" VALUE="TAPES">
    <INPUT TYPE="submit" NAME= "which_button" VALUE="Audio">
    <INPUT TYPE="submit" NAME= "which_button" VALUE="Video">
    <B> <TT> </TT>MP Modules <TT> </TT><B>
</OL>
<P>
or search for: <INPUT NAME="search" SIZE=35 VALUE="">

<INPUT TYPE="submit" NAME="which_button" VALUE="Search">

<INPUT TYPE="hidden NAME="cart" VALUE="AT&T GIS NCR 53C15:}  252

<HR>

[ <A HREF="/dx/index.html">DriverExpress Home</A>
[ <A HREF="/dx/html/browse.html> Driver Reference Information</A>
[ <A HREF="/dx/html/feedback.html"> Feedback</A>
[ <A HREF="/dx/html/help.html" Help</A>]

<P>

<H6 <A HREF="/dx/html/SMICopyright.html>Copyright<A>
&#169 1995 Sun Microsystems, Inc., 2550 Garcia Ave., Mtn.View, CA 94043-1100
USA. Allrights reserved. </H6>

</BODY>
</HTML>
```

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 20 1881

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	PROCEEDINGS OF THE FIRST USENIX WORKSHOP OF ELECTRONIC COMMERCE, PROCEEDINGS OF THE FIRST USENIX WORKSHOP OF ELECTRONIC COMMERCE, NEW YORK, NY, USA, 11-12 JULY 1995, 1995, BERKELEY, CA, USA, USENIX ASSOC, USA, pages 147-154, XP000579443 HAUSER R ET AL: "Generic extensions of WWW browsers" * the whole document * ---	1,10	G06F17/30
A	INTERNET RESEARCH, 1996, MCB UNIVERSITY PRESS, UK, vol. 6, no. 1, ISSN 1066-2243, pages 81-91, XP000670632 ROWLEY J: "Retailing and shopping on the Internet" * the whole document * ---	1,10	
A	DIRECT MARKETING, FEB. 1995, USA, vol. 57, no. 10, ISSN 0012-3188, pages 23-26, XP000670672 FRIED-CASSORLA A: "Successful marketing on the Internet: a user's guide" * the whole document * -----	1,10	TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	1 August 1997	Katerbau, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)